

CPE — Parte 1

Vetores (*Arrays*)

Departamento de Engenharia Elétrica – UnB

Roteiro

- 1 Introdução
- 2 Definição
- 3 Exemplos
- 4 Matrizes
 - Definição
 - Exemplos

Como armazenar 3 notas?

```
float nota1, nota2, nota3;  
  
cout << "Nota do aluno 1: ";  
cin >> nota1;  
cout << "Nota do aluno 2: ";  
cin >> nota2;  
cout << "Nota do aluno 3: ";  
cin >> nota3;
```

Como armazenar 100 notas?

```
float nota1, nota2, nota3, /* .... */ nota100;

cout << "Nota do aluno 1: ";
cin >> nota1;
cout << "Nota do aluno 2: ";
cin >> nota2;

/* ... */

cout << "Nota do aluno 100: ";
cin >> nota100;
```

Vetores (*Arrays*) — Definição

Coleção de variáveis do mesmo tipo referenciada por um nome comum.
(Herbert Schildt - C: The Complete Reference)

- acesso por meio de **índice**
- posições contíguas na memória
- tamanho pré-definido
- índices fora dos limites podem causar comportamento anômalo do código

Como declarar um vetor

<tipo> identificador [*<número de posições>*];

- A *primeira posição* de um vetor sempre tem índice 0.
- A *última posição* de um vetor tem índice *<número de posições> - 1*.

Exemplo

```
float notas[100];
```

Repare que notas é como se tivéssemos 100 variáveis do tipo float.

Usando um vetor

Exemplo - atribuição

```
a = identificador [<posição>];
```

- Pode-se substituir uma variável de um determinado tipo por **um único** elemento de um determinado vetor.
- Este elemento se comporta como uma variável: retorna o seu valor como uma expressão e pode ter valores atribuídos.

Exemplo

```
a = nota[10];  
nota[5] = 9.5;
```

Vetores - Organização na memória

- `int d;`
`int vetor[5];`
`int f;`

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
Valor	?	?	?	?	?	?	?

Vetores - Organização na memória

- Ao executar `vetor[3]=10;`

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
Valor	?	?	?	?	10	?	?

Vetores - Organização na memória

- O que ocorre se digitar os comandos:
vetor[5]=5;
vetor[-1]=1;

Vetores - Organização na memória

- Ao executar
`vetor[3]=10;`
`vetor[5]=5;`
`vetor[-1]=1;`

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
Valor	1	?	?	?	10	?	5

Questões importantes sobre vetores

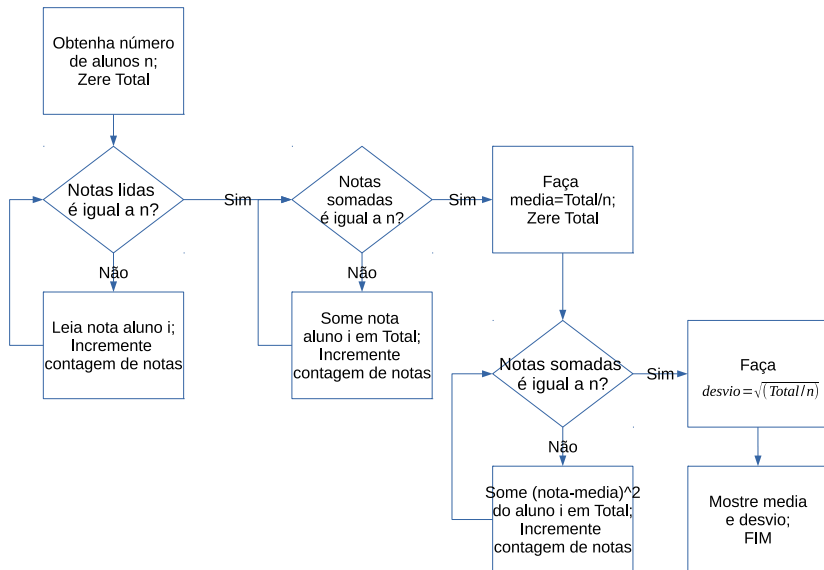
- O tamanho do vetor é **pré-definido** (ou seja, após a compilação o tamanho não pode ser mudado).
- Índices fora dos limites podem causar comportamento anômalo do código.
- Veja exemplo em `limites.cpp`

Como armazenar n (≤ 100) notas?

Crie um algoritmo que leia uma lista de notas (máximo 100), calcule e exiba a média e o seu desvio-padrão.

$$m = \frac{1}{n} \sum_{i=1}^n x_i$$
$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - m)^2}$$

Como armazenar n (≤ 100) notas?



Como armazenar n (≤ 100) notas?

```

1  #include <iostream>
2  #include <cmath>
3  #include <iomanip>
4  using namespace std;
5
6  int main() {
7      float nota[100], media=0, desvpad=0;
8      int n, i;
9
10     cout << "Numero de alunos: ";
11     cin >> n;
12     for (i = 0; i < n; i++) { //leitura das notas
13         cout << "Nota do aluno " << i+1 << ": ";
14         cin >> nota[i];
15     }
16
17     for (i = 0; i < n; i++) //calcula da media
18         media = media + nota[i];
19     media = media / n;
20
21     for (i = 0; i < n; i++) //calcula do desvio padrao
22         desvpad = desvpad + (nota[i]-media)*(nota[i]-media);
23     desvpad = sqrt(desvpad/n);
24
25     cout << "Nota media = " << setprecision(1) << fixed
26         << media << "\nDesvio padrao = " << desvpad << endl;
27     return 0;
28 }

```

Polinômios

Leia os coeficientes a_0, a_1, \dots, a_k , onde $k \leq 25$, e exiba a expressão completa do referido polinômio de ordem k

$$a_k x^k + a_{k-1} x^{k-1} + \dots + a_1 x + a_0$$

Polinômios

```

1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  int main() {
6      float coef[26];
7      int grau, i;
8
9      cout << "Grau do polinomio (grau maximo = 25): ";
10     cin >> grau;
11     while (grau < 0 || grau > 25) {//digitou fora do limite, repete a leitura
12         cout << "Grau invalido, digite outra vez: ";
13         cin >> grau;
14     }
15     for (i = grau; i >= 0; i--) { //leitura dos coeficientes
16         cout << "coeficiente de x^" << i << ": ";
17         cin >> coef[i];
18     }
19     cout << setprecision(1) << fixed << coef[grau] << "x^" << grau;
20     for (i = grau - 1; i >= 0; i--) { //impressao do polinomio
21         if (coef[i] != 0)
22             if (coef[i] >= 0)
23                 cout << " + " << coef[i] << "x^" << i;
24             else
25                 cout << " - " << -coef[i] << "x^" << i;
26     }
27     cout << endl;
28     return 0;
29 }

```

Busca em vetores

- Um problema comum quando se manipula vetores é encontrar um elemento com um determinado valor.
- A forma direta de se resolver é percorrer da posição inicial até a final todos os elementos do vetor, até achar o valor desejado → **Busca Linear ou Sequencial**.

Busca sequencial por um valor

```
1 //Procura um elemento em uma lista de valores – busca linear.
2 #include <iostream>
3
4 using namespace std;
5
6 int main() {
7     int lista[10], chave;
8     int i;
9
10    cout << "Digite uma lista de 10 numeros inteiros: ";
11    for(i=0; i<10; i++)
12        cin >> lista[i];
13
14    cout << "Digite um numero a procurar na lista: ";
15    cin >> chave;
16
17    //laco de busca – veja que e somente 1 linha de comando
18    for (i = 0; lista[i] != chave && i<10; i++);
19
20    if (i<10) //se laco for parou antes do 10, significa que encontrou chave
21        cout << "O elemento " << chave << " esta presente na lista\n";
22    else //caso contrario, nao encontrou a chave
23        cout << "O elemento " << chave << " nao esta presente na lista\n";
24
25    return 0;
26 }
```

Busca sequencial por um valor - outra implementação

```
1  //Procura um elemento em uma lista de valores - busca linear.
2  #include <iostream>
3
4  using namespace std;
5  int main() {
6      int lista[10], chave;
7      int i;
8      bool achou;
9
10     cout << "Digite uma lista de 10 numeros inteiros: ";
11     for(i=0; i<10; i++)
12         cin >> lista[i];
13
14     cout << "Digite um numero a procurar na lista: ";
15     cin >> chave;
16
17     achou = false;
18     //laco de busca
19     for (i = 0; !achou && i<10; i++){
20         if(lista[i]==chave)
21             achou = true;
22     }
23
24     if (achou)
25         cout << "O elemento " << chave << " esta presente na lista\n";
26     else
27         cout << "O elemento " << chave << " nao esta presente na lista\n";
28
29     return 0;
30 }
```

Matrizes

- Utilizando vetores, criamos um programa que lê as notas de uma prova para um conjunto de alunos e então calcula a média da turma.
- Agora queremos ler as notas de 4 provas para cada aluno e então calcular a média do aluno e a média da classe. O tamanho máximo da turma é de 50 alunos.

solução

Criar 4 vetores cada um com 50 posições. E então ler as respectivas informações.

```
float nota0[50],nota1[50],nota2[50],nota3[50];
```

Matrizes

- Agora suponha que estamos trabalhando com no máximo 100 provas e 100 alunos. Seria muito cansativo criar 100 vetores e atribuir 100 nomes diferentes. (Parece que esse problema não tem fim !!!).
- Para resolver esse problema podemos utilizar **matrizes**. Uma matriz é um vetor (ou seja, um conjunto de variáveis de mesmo tipo) que possui **duas ou mais dimensões**.

Declarando uma matriz

```
<tipo> nome_da_matriz [<linhas>] [<colunas>]
```

- Uma matriz possui *linhas* \times *colunas* variáveis do tipo **<tipo>**
- As linhas são numeradas de 0 a *linhas* $- 1$
- As colunas são numeradas de 0 a *colunas* $- 1$

Exemplo de declaração de matriz

```
int matriz[4][4];
```

	0	1	2	3
0				
1				
2				
3				

Declarando uma matriz de múltiplas dimensões

```
<tipo> nome_da_matriz [< dim1 >] [< dim2 >] ... [< dimN >]
```

- Essa matriz possui $dim_1 \times dim_2 \times \dots \times dim_N$ variáveis do tipo `<tipo>`
- Cada dimensão é numerada de 0 a $dim_i - 1$

Acessando uma matriz

- Em qualquer lugar onde você escreveria uma variável no seu programa, você pode usar um elemento de sua matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz [1] [10]` — Refere-se a variável na 2ª linha e na 11ª coluna da matriz.

- Lembre-se sempre:** o compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

Exemplo: lendo e escrevendo uma matriz

```
1 #include <iostream>
2 using namespace std;
3
4 int main () {
5     int matriz[4][4];
6     int i, j;
7
8     /*Leitura*/
9     for (i = 0; i < 4; i++) //para todas as linhas
10         for (j = 0; j < 4; j++) { //para todas as colunas de cada linha
11             cout << "Matriz[" << i << "][" << j << "]: ";
12             cin >> matriz[i][j];
13         }
14
15     /*Escrita*/
16     for (i = 0; i < 4; i++) { //cada linha
17         for (j = 0; j < 4; j++) //cada coluna de cada linha
18             cout << matriz[i][j] << "\t";
19         cout << endl;
20     }
21
22     return 0;
23 }
```

Exemplo: matriz e sua transposta

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      int mat[3][3];
7      int i = 0, j = 0, k = 0;
8
9      for (i = 0; i < 3; i++) //leitura dos elementos da matriz
10         for (j = 0; j < 3; j++){
11             cout << "Digite o valor da posicao (" << i << ", " << j << "): ";
12             cin >> mat[i][j];
13         }
14
15     cout << "Matriz\n";
16     for (i = 0; i < 3; i++){ //exibicao da matriz original
17         for (j = 0; j < 3; j++){
18             cout << mat[i][j] << "\t";
19             cout << "\n";
20         }
21
22     cout << "Transposta\n";
23     for (i = 0; i < 3; i++){ //exibicao da matriz transposta
24         for (j = 0; j < 3; j++){
25             cout << mat[j][i] << "\t";
26             cout << "\n";
27         }
28     return 0;
29 }
```

Dilbert

